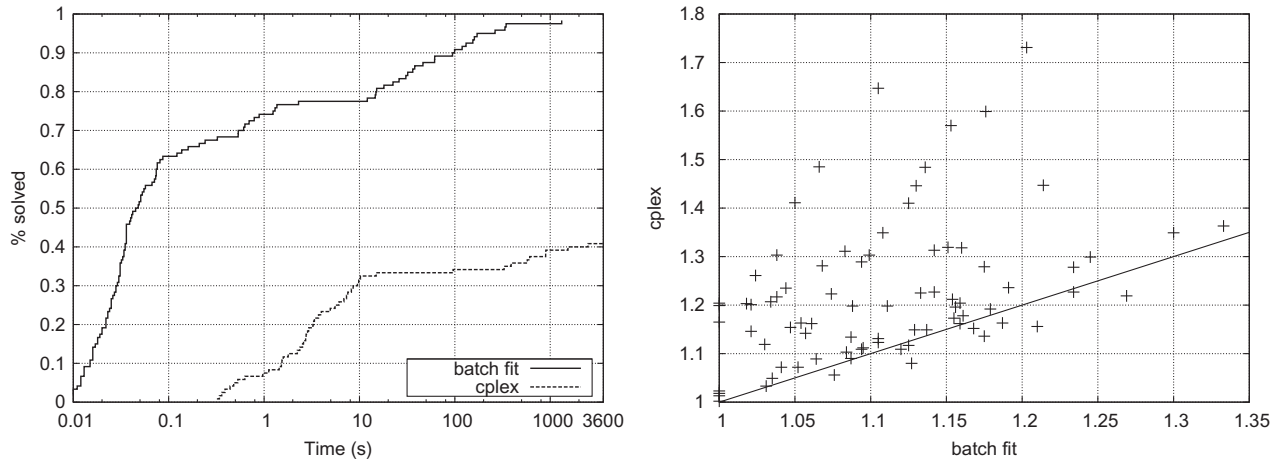

(a) Time comparison ($n \leq 50$). (b) Gap comparison ($n > 50$).

Fig. 5. Comparison to the mathematical formulation.

Their algorithm has been tested on instances presented in Section 6.1 with less than 50 jobs and a time limit of 3600 seconds (1 hour). All their experiments have been conducted on a Pentium 2.66 GHz with 1 GB of RAM. Detailed results on benchmark instances are not mentioned. A 55% and 8% of instances of size $n = 20$ and $n = 50$ are solved by the branch-and-price procedure within one hour. Although their branch-and-price is more efficient than the mathematical formulation, it is less efficient than our constraint programming which solves optimally all instances of size $n = 20$ with an average solution time lower than 1 second, and 95% of instances of size $n = 50$ with an average solution time lower than 100 seconds.

7. Conclusion

We have presented a constraint programming approach to minimize the maximal lateness for a batch processing machine on which a finite number of jobs with non-identical sizes must be scheduled. This approach exploits a new optimization constraint based on a relaxed problem which applies cost-based domain filtering rules and the search is enhanced with dedicated branching heuristics. Computational results demonstrate the positive effect of each component and give better solutions with computation times that are orders of magnitude lower than a mathematical formulation or a branch-and-price based on bin packing and sequencing models.

In further research, we will apply our approach to problems with completion time related measures. In addition, subsequent research topics include the study of parallel batching machines and additional constraints, for instance job release dates that remain incompatible with our approach.

Appendix A. Short description of the `pack` constraint

In this appendix, we briefly describe our implementation of the global constraint `pack` originally proposed by Shaw (2004) for the bin-packing problem. Let recall that this constraint replaces the channeling constraints between assignment variables ($\forall j \in J$, $B_j = k \iff j \in J_k$) and enforces the consistency between assignments and loads $\left(\forall k \in K,\ \sum_{j \in E_k} s_j = S_k\right)$. Furthermore, `pack` propagates the redundant constraint specifying that the sum of the bin loads is equal to the sum of the item sizes $\left(\sum_J s_j = \sum_K S_k\right)$. First the constraint performs the propagation of typical model which corresponds to Constraints (6) and (7) of the mathematical formulation (see Section 6.4). In addition, it uses propagation rules incorporating knapsack-based reasoning, as well as a dynamic lower bound on the number of bins required.

The additional constraint propagation rules are based upon treating the simpler problem of packing items into a single bin. That is, given a bin, can we find a subset of items that when packed in the bin would bring the load in a given interval? This problem is a type of knapsack or subset sum problem (Kellerer et al., 2004). Proving that there is no solution to this problem for any bin would mean that search could be pruned. If an item appears in every set of items that can be placed in the bin, we can commit it to the bin. Conversely, if the item never appears in such a set, we can eliminate it as a candidate item. So, if no legal packing can attain a given load, it cannot be a legal load for the bin. Shaw (2004) proposed efficient algorithms which do not depend on item sizes or bin capacity, but which do not in general achieve generalized arc consistency on the subset sum problem as opposed to Trick (2003). Next, they adapt bounding procedures to partial assignments. The idea is to transform a partial assignment into a new bin packing instance and to apply a bounding procedure to this new instance. Our implementation uses the lower bound L^{1D}_{CCM} (Carlier et al., 2007), which dominates the lower bound L^{1D}_{MV} originally used.

Our implementation differs also from the one of Shaw (2004) by the addition of variables S_k and M. Indeed, variables S_k helps to express additional constraints and were maintained internally by Shaw (2004). The variable M represents the objective of the bin packing problem and its value can be restricted by other constraints.

Shaw (2004) demonstrated that this global constraint can cut search by orders of magnitude. Additional comparisons showed that the global constraint coupled with a standard packing algorithm based on *complete decreasing* can significantly outperform Martello and Toth's (1990) procedure.

Appendix B. Algorithm for cost-based domain filtering of assignments

We recall that a simple algorithm can compute marginal costs from scratch for each possible assignment with an overall complexity of $O(n^3)$. In this section, we describe an $O(nm)$ version of this algorithm (cf. Algorithm 1) based on the incremental computation of marginal costs. The principle consists in exploiting the formula below for quick computation of marginal costs by distinguishing several cases. For instance, the assignment of a job to any empty batch leads to the same marginal cost. In fact, the com-